

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-207224

(43)Date of publication of application : 28.07.2000

(51)Int.Cl.

G06F 9/45

G06F 12/08

(21)Application number : 11-008234

(71)Applicant : HITACHI LTD

(22)Date of filing : 14.01.1999

(72)Inventor : KUSHIMA ICHIRO

(54) SOFTWARE PREFETCHING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To prevent prefetched data from being expelled from a cache before being used, when data which is not in the innermost loop is referred to.

SOLUTION: A loop (605 to 607) immediately prior to array reference A (i) is taken out, when a number (d) of predictive execution cycles of the whole loop (605 to 607) is larger than a number (k) (memory latency) of cycles needed to prefetch data, a number (e) of predictive execution cycles per loop (605 to 607) is found, and a loop frequency $\beta = (k/e)$ corresponding to the memory latency is calculated. Then the loop (605 to 607) is divided into two loops, i.e., a former half loop (624 to 627) for repetition of 1 to $(M-\beta)$ times and a latter half loop (628 to 630) for repetition of $(M-\beta+1)$ to M times and a prefetch instruction (627) is put between the former half loop (624 to 627) and the latter half loop (628 to 630).

```

real A(M),B(N,M)      (601)
a=0                    (602)
do i=1,N                (603)
  a=0                  (604)
  do j=1,M              (605)
    a=a+B(j,i)         (606)
  enddo                (607)
enddo                  (608)

```

(a)

	701	702	703	704	705	706	707
ループ 番号	1	2	3	4	5	6	7
ループ 範囲	1	2	3	4	5	6	7
ループ 回数	1	2	3	4	5	6	7

(b)

```

real A(M),B(N,M)      (620)
a=0                    (621)
do i=1,N              (622)
  a=0                  (623)
  do j=1,M-β          (624)
    a=a+B(j,i)         (625)
  enddo                (626)
  prefetch(A(i))       (627)
  do j=β+1,M           (628)
    a=a+B(j,i)         (629)
  enddo                (630)
enddo                  (631)

```

(c)

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

THIS PAGE BLANK (USPTO)

Copyright (C); 1998,2003 Japan Patent Office

Copyright (C); 1998,2003 Japan Patent Office

THIS PAGE BLANK (USPTO)

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2000-207224

(P2000-207224A)

(43)公開日 平成12年7月28日(2000.7.28)

(51)Int.Cl.⁷

識別記号

F I

テーマコード(参考)

G 0 6 F 9/45

C 0 6 F 9/44

3 2 2 C 5 B 0 0 0

12/08

12/08

D 5 B 0 8 1

審査請求 未請求 請求項の数12 O L (全 13 頁)

(21)出願番号 特願平11-8234

(22)出願日 平成11年1月14日(1999.1.14)

(71)出願人 000003108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 久島 伊知郎

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(74)代理人 10008/170

弁理士 富田 和子

Fターム(参考) 5B005 JJ13 MM01

5B081 CC30 CC41

(54)【発明の名称】 ソフトウェアプリフェッチ方法

(57)【要約】

【課題】 最内側ループ中にあるデータの参照に対し、プリフェッチしたデータが使用される前にキャッシュから追い出されるのを防止する。

【解決手段】 配列参照A(i)の直前にあるループ(605~607)を取り出し、ループ(605~607)全体の予測実行サイクル数dがデータをプリフェッチするのに必要なサイクル数(メモリレイテンシ)kより大きければ、ループ(605~607)のループ1回あたりの予測実行サイクル数eを求め、メモリレイテンシに相当するループ回数 $\beta = \lceil k/e \rceil$ を計算する。そして、ループ(605~607)を、1~(M- β)回の繰り返しを実行する前半ループ(624~627)と、(M- β +1)~M回の繰り返しを実行する後半ループ(628~630)の2つのループに分割し、前半ループ(624~627)と後半ループ(628~630)の間に、プリフェッチ命令(627)を挿入する。

図5

```

real A(N),B(M,N)      (601)
s=0                    (602)
do i=1,N               (603)
  t=0                  (604)
  do j=1,M              (605)
    t=t+B(j,i)         (606)
  enddo                (607)
  s=t+A(i)             (608)
enddo                  (609)

```

(a)

ループ番号	ループ変数	繰り返し回数	内側ループ	内側ループ	内側ループ	予測実行サイクル
1	i	N	なし	なし	なし	c*N
2	j	M	なし	なし	なし	c

(b)

```

real A(N),B(M,N)      (620)
s=0                    (621)
do i=1,N               (622)
  t=0                  (623)
  do j=1,M- $\beta$           (624)
    t=t+B(j,i)         (625)
  enddo                (626)
  prefetch(A(i))       (627)
  do j=M- $\beta$ +1,M          (628)
    t=t+B(j,i)         (629)
  enddo                (630)
  s=t+A(i)             (631)
enddo                  (632)

```

(c)

【特許請求の範囲】

【請求項1】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおいて、
 プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、
 当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求め、
 前記第2のループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、
 前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とするソフトウェアプリフェッチ方法。

【請求項2】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおいて、
 プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、
 当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求め、
 前記第2のループの、最後から β 回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とするソフトウェアプリフェッチ方法。

【請求項3】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおいて、
 プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、
 当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求め、
 前記ループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、
 前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とするソフトウェアプリフェッチ方法。

【請求項4】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおいて、
 プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、
 当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数

であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求め、
 前記ループの、最後から β 回目の繰り返しのときに、プリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とするソフトウェアプリフェッチ方法。

【請求項5】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置であってプリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求める手段と、

前記第2のループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、

前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とするコンパイル装置。

【請求項6】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置であって、

プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求める手段と、

前記第2のループの、最後から β 回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とするコンパイル装置。

【請求項7】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置であって、
 プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求める手段と、

前記ループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、

前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とするコンパイル装置。

【請求項8】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置であって、

て、
プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求める手段と、
前記ループの最後から β 回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とするコンパイル装置。

【請求項9】 ソースコードを読み込んで、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置において、
前記ソースコードが、第1のループを有し、更に、当該第1のループの内側にプリフェッチの対象とされるメモリ参照及び第2のループを有する場合、
前記オブジェクトコードは、
前記第2のループに対応する前半ループ及び後半ループと、
前半ループと後半ループの間にプリフェッチ命令とを有し、
前記後半ループの繰り返し回数は、データをプリフェッチするのに要するサイクル数に相当する回数であることを特徴とするコンパイル装置。

【請求項10】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラであって、
プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求めるステップと、
前記第2のループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割するステップと、
前半ループと後半ループの間にプリフェッチ命令を挿入するステップとを備えることを特徴とするコンパイラを記録した記録媒体。

【請求項11】 オブジェクトコードを記録した記録媒体であって、
前記オブジェクトコードは、
1つのループ処理に相当する前半ループ及び後半ループと、
後半ループのうしろに、メモリ参照をする命令と、
前半ループと後半ループの間に、前記メモリ参照についてのプリフェッチ命令とを有し、
前記後半ループの繰り返し回数は、前記オブジェクトコ

ードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する回数であることを特徴とする記録媒体。

【請求項12】 オブジェクトコードを記録した記録媒体であって、
前記オブジェクトコードは、
ループと、
当該ループのうしろに、メモリ参照をする命令と、
前記ループの最後から β 回目の繰り返しのときに、前記メモリ参照についてのプリフェッチ命令を実行するコードとを有し、
前記 β は、前記オブジェクトコードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する前記ループの繰り返し回数であることを特徴とする記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、計算機利用技術におけるコンパイル方法に関し、特に、プリフェッチ命令をコードに挿入することによりキャッシュミスの削減を図るソフトウェアプリフェッチ方法に関する。

【0002】

【従来の技術】近年、命令レベルの並列度や動作周波数の向上によってマイクロプロセッサの性能は飛躍的に向上している。これに対して、計算機の主記憶を構成するDRAMの性能向上はプロセッサ性能の向上に比べて低いレベルに留まっている。この結果、主記憶参照に要するサイクル数（メモリレイテンシ）は増加する傾向にある。

【0003】これに対して、従来のシステムでは、データ参照の局所性に着目して、主記憶と比較して高速な少容量のメモリ（キャッシュ）をプロセッサと主記憶の間に配置し、最近参照したデータをキャッシュ上に置くことによって、主記憶参照の回数を減らし、全体として主記憶参照に要する時間を短縮するという方式をとっていた。ところが、数値計算処理などの大規模なデータを使用する計算では、データ参照の局所性が低いためキャッシュミスが多発し、主記憶参照によって生じる待ち時間によってプログラムの性能が大幅に低下してしまうという問題があった。

【0004】このような大規模データに対するキャッシュミスの増加に対処するため、例えば、文献「T.C.Mowry 他, "Design and Evaluation of a Compiler Algorithms for Prefetching", Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.62-73, 1992」に記載されているように、主記憶からキャッシュへデータを先行的に読み出す特別な命令（プリフェッチ命令）をプロセッサの命令セットに用意し、コンパイラによってプログラム中にプリフェッチ命令を挿入す

る方式（ソフトウェアプリフェッチ）が検討されている。このプリフェッチ命令を利用すれば、後続のループ繰り返し等で使用するデータを予め主記憶からキャッシュへ読み込みながら、同時に別の演算を行うことができ、これによって主記憶参照による待ち時間を隠蔽することができる。

【0005】図11は、従来のソフトウェアプリフェッチ方法の例を示す図である。同図(a)は、コンパイル処理の対象となるFORTRANで記述されたソースプログラムを表す。

【0006】ここでは、同図(a)に示すプログラムにおいて、配列要素A(i)の参照(504)について、プリフェッチ命令を挿入する場合を考える。同図(a)に示すように、i番めのループ繰り返して配列要素A(i)を参照するようなループ(503~505)があった場合、まず、ループの1回の繰り返しにかかる実行サイクル数cを見積もり、データをメモリからキャッシュに移動するのに要するサイクル数(メモリレイテンシ)Lをcで割った値 $\alpha = [L/c]$ （ここで、[]は、小数点以下切り上げを表す記号）を計算する。そして、i回目の繰り返して配列要素A(i+ α)をプリフェッチする命令を配列要素A(i)の直前に挿入する。

【0007】同図(b)は、プリフェッチ命令(509)を挿入した後のプログラムをソースコードイメージで表したものである。このように α 回後の繰り返して参照するデータに対して予めプリフェッチ命令を発行しておくことにより、 $L (= c * \alpha)$ サイクル後にそのデータを参照するときは当該データが既にキャッシュに到着しているため、キャッシュヒットとなる。これによりキャッシュミスによるプログラム実行性能の低下が防止できる。

【0008】

【発明が解決しようとする課題】上記文献に示されているソフトウェアプリフェッチ方法では、プリフェッチしようとする配列の参照が最内側ループ（すなわち、その内側に他のループを含まないループ）にあるような単純な場合のみを仮定している。

【0009】前述したソフトウェアプリフェッチ方法を、プリフェッチの対象とする配列参照を有するループの内側にさらに別のループがある場合に適用すると、プリフェッチしたデータが、内側ループの実行によりキャッシュから追い出されてしまう恐れがある。

【0010】図12は、このような場合のソフトウェアプリフェッチ方法の例を示す図である。同図(a)は、コンパイル処理の対象となるFORTRANで記述されたソースプログラムを示す。同図(a)に示したプログラムにおいて、配列参照A(i)(608)をプリフェッチの対象とした場合、同図(b)のようにプリフェッチ命令(617)が挿入される。つまり、ループのi番めの繰り返して配列要素A(i+ α)をプリフェッチし

ている。しかし、このプリフェッチ命令の発行の後、次以降のiの繰り返して、内側ループ(614~616)の実行がされるので、そこで参照されるデータB(j, i)と、データA(i+ α)との間でキャッシュラインの競合が発生すると、プリフェッチによってキャッシュにロードされたデータA(i+ α)がキャッシュから追い出されてしまい、実際に値を使用するときはキャッシュミスを起こすことになり、プリフェッチの効果がなくなることになる。

【0011】本発明の目的は、最内側ループ中にないデータの参照に対して、プリフェッチの効果をを得ることができるソフトウェアプリフェッチ方法を提供することにある。

【0012】

【課題を解決するための手段】本発明に係るソフトウェアプリフェッチ方法は、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおけるソフトウェアプリフェッチ方法である。そして、プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求め、前記第2のループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とする。

【0013】この場合、プリフェッチ命令とメモリ参照との間に実行される第2のループは、 β 回しか回らず、 β は一般に分割前の元のループ繰り返し回数に比べて小さいので、その間に参照されるデータは一般に小さく、プリフェッチしたデータがキャッシュから追い出される可能性は少なくなる。また、ループの β 回の実行サイクル数は、メモリレイテンシ、すなわち、データをメモリからプリフェッチするのに要するサイクル数に相当するので、データを参照するときにプリフェッチデータの到着が間に合っていないということがない。

【0014】なお、第2のループが多重ループになっている場合には、第2のループを分割してできた後半ループの内側ループをさらに分割することにより、より適当な位置にプリフェッチ命令を挿入するようにしてもよい。

【0015】また、本発明に係る第2のソフトウェアプリフェッチ方法は、プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰

り返し回数 β を求め、前記第2のループの最後から β 回目の繰り返しの際にプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とする。

【0016】また、本発明に係る第3のソフトウェアプリフェッチ方法は、プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求め、前記ループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とする。

【0017】また、本発明に係る第4のソフトウェアプリフェッチ方法は、プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求め、前記ループの最後から β 回目の繰り返しの際にプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とする。

【0018】本発明に係るコンパイル装置は、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成する装置である。そして、プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求める手段と、前記第2のループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とする。

【0019】また、本発明に係る第2のコンパイル装置は、プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求める手段と、前記第2のループの最後から β 回目の繰り返しの際にプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とする。

【0020】また、本発明に係る第3のコンパイル装置

は、プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求める手段と、前記ループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とする。

【0021】また、本発明に係る第4のコンパイル装置は、プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求める手段と、前記ループの最後から β 回目の繰り返しの際にプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とする。

【0022】本発明に係る第5のコンパイル装置は、ソースコードが、第1のループを有し、更に、当該第1のループの内側に、プリフェッチの対象とされるメモリ参照及び第2のループを有する場合、前記第2のループに対応する前半ループ及び後半ループと、前半ループと後半ループの間にプリフェッチ命令とを有し、前記後半ループの繰り返し回数は、データをプリフェッチするのに要するサイクル数に相当する回数であるオブジェクトコードを生成することを特徴とする。

【0023】また、本発明に係る第6のコンパイル装置は、ソースコードが、第1のループを有し、更に、当該第1のループの内側に、プリフェッチの対象とされるメモリ参照及び第2のループを有する場合、前記第2のループの最後から β 回の繰り返しの際にプリフェッチ命令を実行するコードを有し、前記 β は、データをプリフェッチするのに要するサイクル数に相当する第2のループの繰り返し回数であるオブジェクトコードを生成することを特徴とする。

【0024】また、本発明に係る第7のコンパイル装置は、ソースコードが、プログラム手続きを有し、更に、当該プログラム手続き内に、プリフェッチの対象とするメモリ参照及びループを有する場合、前記ループに対応する前半ループ及び後半ループと、前半ループと後半ループの間にプリフェッチ命令とを有し、前記後半ループの繰り返し回数は、データをプリフェッチするのに要するサイクル数に相当する回数であるオブジェクトコードを生成することを特徴とする。

【0025】また、本発明に係る第8のコンパイル装置は、ソースコードが、プログラム手続きを有し、更に、当該プログラム手続き内に、プリフェッチの対象とする

メモリ参照及びループを有する場合、前記ループの最後から β 回目の繰り返しのときに、プリフェッチ命令を実行するコードを有し、前記 β は、データをプリフェッチするのに要するサイクル数に相当する前記ループの繰り返し回数であるオブジェクトコードを生成することを特徴とする。

【0026】本発明に係る第1の記録媒体は、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラを記録した記録媒体である。そして、当該コンパイラは、プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求めるステップと、前記第2のループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割するステップと、前半ループと後半ループの間にプリフェッチ命令を挿入するステップとを備えることを特徴とする。

【0027】また、本発明に係る第2の記録媒体は、第2のコンパイラを記録した記録媒体である。この第2のコンパイラは、プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数 β を求めるステップと、前記第2のループの最後から β 回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入するステップとを備えることを特徴とする。

【0028】また、本発明に係る第3の記録媒体は、第3のコンパイラを記録した記録媒体である。この第3のコンパイラは、プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求めるステップと、前記ループを、終わりの β 回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割するステップと、前半ループと後半ループの間にプリフェッチ命令を挿入するステップとを備えることを特徴とする。

【0029】また、本発明に係る第4の記録媒体は、第4のコンパイラを記録した記録媒体である。この第4のコンパイラは、プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し1回あたりの実行サイクル

数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数 β を求めるステップと、前記ループの最後から β 回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入するステップとを備えることを特徴とする。

【0030】本発明に係る第5及び第6の記録媒体は、オブジェクトコードを記録した記録媒体である。そして、第5の記録媒体において、前記オブジェクトコードは、1つのループ処理に相当する前半ループ及び後半ループと、後半ループのうしろにメモリ参照をする命令と、前半ループと後半ループの間に、前記メモリ参照についてのプリフェッチ命令とを有し、前記後半ループの繰り返し回数は、前記オブジェクトコードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する回数であることを特徴とする。

【0031】また、第6の記録媒体において、前記オブジェクトコードは、ループと、当該ループのうしろに、メモリ参照をする命令と、前記ループの最後から β 回目の繰り返しのときに、前記メモリ参照についてのプリフェッチ命令を実行するコードとを有し、前記 β は、前記オブジェクトコードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する前記ループの繰り返し回数であることを特徴とする。

【0032】なお、前記記録媒体は、コンピュータが読みとり可能な任意の記録媒体であり、例えば、半導体メモリ、フロッピーディスク、CD-ROM、磁気ディスク、光磁気ディスク等が該当する。

【0033】

【発明の実施の形態】以下、本発明の実施の形態について図面を参照して詳細に説明する。

【0034】図1は、本発明によるコンパイラが稼動する計算機システムの構成を示す図である。同図に示すように、本計算機システムは、CPU101、ディスプレイ装置102、キーボード103、主記憶装置104、及び外部記憶装置105を備える。

【0035】CPU101は、主記憶装置104にロードされたコンパイラ108を実行することにより、コンパイル処理を行う。

【0036】ディスプレイ装置102は、例えば、CRT表示装置や液晶表示装置等で構成され、コンパイルの終了やエラー等をユーザに知らせる各種メッセージ等を表示する。

【0037】キーボード103は、ユーザが、ソースプログラム106のコーディングを行ったり、コンパイラ108を起動するコマンドを入力する際等に使われる入力装置である。

【0038】主記憶装置104は、例えば、DRAM等の半導体メモリで構成され、コンパイラ108や、コンパイル処理過程で必要となる中間コード109およびループ表110等を格納する。

【0039】外部記憶装置105は、例えば、磁気ディスク等で構成され、ソースプログラム106、オブジェクトプログラム107等を格納する。

【0040】図1に示したような計算機システムにおいて、ユーザがキーボード103等を使ってコンパイル処理の実行を指示すると、コンパイラ108は、ソースプログラム106を読み込んで、コンパイル処理を行い、対応するオブジェクトプログラム107を生成する。

【0041】図2は、コンパイラ108のコンパイル処理の流れを示すフローチャートである。コンパイラ108は、まず、構文解析処理を行う(S201)。ここでは、ソースプログラム106を読み込み、字句解析、構文解析、意味解析等を行って、コンパイラ内部で処理可能な中間コード109を作成する。構文解析処理の詳細については、例えば、「エイホ、セシィ、ウルマン著：コンパイラI（サイエンス社、1990年）」の30頁～74頁に記載されている。

【0042】次に、コンパイラ108はループ解析処理を行う(S202)。ここでは、プログラムに含まれるループの集合や各ループの間の関係等が求められ、ループ表110に記録される。ループ表110に記録された情報は、後のコンパイル処理において適宜参照される。ループ解析処理の詳細については、例えば、「エイホ、セシィ、ウルマン著：コンパイラII（サイエンス社、1990年）」の734頁～737頁に記載されている。

【0043】次に、コンパイラ108は、各ループに対して、ソフトウェアプリフェッチ処理を行う(S203～S205)。そのため、まず、未処理のループがあるか否かを調べ(S203)、未処理のループがあれば(S203:yes)、未処理のループを1つ取り出して、処理対象ループnとする(S204)。そして、処理対象ループn中の配列参照に対してソフトウェアプリフェッチ処理を行う(S205)。このソフトウェアプリフェッチ処理の詳細については後述する。ソフトウェアプリフェッチ処理が終了すると、他の未処理ループについての処理に進む(S203)。

【0044】そして、すべてのループについて処理を終了すると(S203:no)、オブジェクトコードを生成し(S206)、コンパイル処理を終了する。オブジェクトコード生成処理の詳細については、例えば、前記「コンパイラII」の624頁～707頁に記載されている。

【0045】図3は、前述したステップS205におけるソフトウェアプリフェッチ処理の流れを示す図である。ここでは、処理対象ループn中の配列参照に対して、適切な位置にプリフェッチ命令を挿入する処理を行

う。

【0046】まず、処理対象ループn中に未処理かつプリフェッチの対象とすべき配列参照があるか否かを調べる(S301)。ある配列参照をプリフェッチの対象とすべきか否かは、例えば、その配列参照のメモリアドレスを予め求めることができるか否か、配列の添字がループ制御変数の線形関数となっているか否か、配列データが既にキャッシュにあるか否か、等に基づいて判断する。

【0047】その結果、未処理かつプリフェッチの対象とすべき配列参照があれば(S301:yes)、その配列参照を1つ取り出して、処理対象配列参照sとする(S302)。次に、変数Lに、メモリレイテンシ（プリフェッチに要するサイクル数）を代入する(S303)。Lに代入する値は、ターゲットマシンによって、適当な値がコンパイラに予め与えられるが、コンパイラ起動時等に、ユーザがパラメータとして与えるようにしてもよい。

【0048】次に、処理対象ループnが最内側ループであるか否かを調べる(S304)。処理対象ループnが最内側ループであるか否かは、例えば、ループ表110を参照して調べる。その結果、処理対象ループnが最内側ループであれば(S304:yes)、従来と同様の方法で、プリフェッチ命令の挿入を行う(S305、S306)。すなわち、まず、処理対象ループnのループ1回あたりの予測実行サイクル数cを求め、データをプリフェッチするのに必要なサイクル数（メモリレイテンシ）Lに相当するループ回数 $\alpha = \lceil L/c \rceil$ を計算する(S305)。そして、処理対象配列参照sの添字式中のループ繰り返し変数iを $i + \alpha$ に置き換えた配列要素をプリフェッチする命令を生成し、処理対象配列参照sの直前に挿入する(S306)。

【0049】一方、処理対象ループnが最内側ループでない場合は(S304:no)、従来とは異なる方法で、処理対象配列参照sのプリフェッチ命令を挿入すべき位置を求める処理を行う(S307)。このステップS307における処理の詳細については、後述する。プリフェッチ命令を挿入すべき位置が求めると、その位置に、処理対象配列参照sをプリフェッチする命令を挿入する(S308)。

【0050】以上の処理S301～S308を繰り返して、すべてのプリフェッチ対象となる配列参照について処理を終了すると(S301:no)、ソフトウェアプリフェッチ処理を終了する。

【0051】次に、ステップS307におけるプリフェッチ命令を挿入すべき位置を求める処理について詳細に説明する。図4は、ステップS307における処理の流れを示す図である。ここでは、処理対象配列参照sのLサイクル前に実行される位置を求める。その際、必要に応じてループの分割を行う。

【0052】まず、処理対象ループn中の処理対象配列参照sの直前にあるループを分割対象ループmとして取り出す(S401)。ここで、処理対象配列参照sの前にループが存在しない場合は、処理対象ループn中の最後のループを分割対象ループmとして取り出す。なお、ここでの処理対象ループnは、最内側ループではないので、処理対象ループn中には、必ずループが存在する。

【0053】次に、変数kの初期値として、Lの値を代入する(S402)。そして、分割対象ループmの全体(全繰返し)の予測実行サイクル数を求め、それをdとする(S403)。分割対象ループmの繰返し数がコンパイル時に不明で、全体の予測実行サイクル数が求まらないときは、 $d \leftarrow \infty$ (例えば、dとしてとりえない特定の値)としておく。

【0054】次に、dがkより大きいかな否かを調べる(S404)。その結果、dがk以下の場合は(S404: no)、現在の分割対象ループmの分割は行わず、分割対象ループmを、現在の分割対象ループmの直前のループに変更し、また、kから現在の分割対象ループmの全体の予測実行サイクル数dを減算する(S410)。なお、分割対象ループmの前にループがない場合は、処理対象ループn中の最後のループを新たな分割対象ループmとして選択する。そして、ステップS403～S404の処理を再度行う。

【0055】一方、dがkより大きければ(S404: yes)、分割対象ループmの分割を行うため、まず、分割対象ループmのループ1回分の予測実行サイクル数eを求め、kに相当する分割対象ループmのループ回数 $\beta = \lceil k/e \rceil$ を計算する(S405)。

【0056】そして、分割対象ループmを、最後の β 回のみを実行する後半ループm2と、その前までの繰返しを実行する前半ループとに分割する(S406)。すなわち、分割対象ループmの繰返し回数をNとしたとき、 $1 \sim (N - \beta)$ 回の繰返しを実行する前半ループm1と、 $(N - \beta + 1) \sim N$ 回の繰返しを実行する後半ループm2の2つのループに分割する。なお、 $\beta = 1$ の場合は、後半ループm2は1回しか実行されないのではループにしなくてもよい。また、ループの分割にあわせて、ループ表110の更新を行う。

【0057】次に、後半ループm2が多重ループであるか否か、すなわち、その内側に他のループを含むループであるかを否かを調べる(S407)。その結果、後半ループm2が多重ループである場合は(S407: yes)、更なるループ分割を行うため、後半ループm2の内側ループを、分割対象ループmとして、ステップS403～S407の処理を再度行う。なお、後半ループm2が複数の(同階層の)内側ループを有する場合は、一番最後のループを分割対象ループmとする。

【0058】一方、後半ループm2が多重ループでなければ(S407: no)、これ以上、ループ分割をする

ことはできないので、その時点での前半ループm1と後半ループm2の間を、プリフェッチ命令を挿入する位置とし(S409)、処理を終了する。

【0059】次に、上述したコンパイラ108の処理を、具体例に基づいて説明する。まず、図5(a)に示したソースプログラム(図12(a)と同じソースプログラム)をコンパイルする場合について説明する。

【0060】図5(a)に示したプログラムのコンパイルが指示されると、コンパイラ108は、ソースプログラムを読み込み、構文解析処理S201を行った後、ループ解析処理S202を行う。

【0061】図5(b)は、このループ解析処理S202によって作成されるループ表110の構成例を示す図である。同図に示すように、ループ表110は、各ループに対して、ループ番号701、ループ繰返し変数702、繰返し回数703、内側ループ704、直前ループ705、直後ループ706、予測実行サイクル707等の情報を格納する。

【0062】図5(a)のプログラムには、ループが2つあり、外側ループ(603～609)には、「1」、内側ループ(605～607)には、「2」というループ番号が付けられている。なお、以下では、ループ番号が「1」のループをループ1、ループ番号が「2」のループをループ2、等と呼ぶ。

【0063】ループ1のループ繰返し変数は「i」、繰返し回数は「N」、内側ループは、ループ「2」、直前ループ、直後ループは、「なし」である。また、ループ2のループ繰返し変数は「j」、繰返し回数は「M」、内側ループ、直前ループ、直後ループはいずれも「なし」である。

【0064】予測実行サイクル数707には、各ループのループ1回あたりの予測実行サイクル数が格納される。この予測実行サイクル数は、ループ内の命令数と各命令の実行サイクル数、内側ループの繰返し回数等から計算される。図5の例では、ループ2のループ1回あたりの予測実行サイクルを「c」と見積り、ループ1のループ1回あたりの予測実行サイクルは、ループ2以外の部分の実行サイクル数を無視して、「 $c * M$ 」と計算している。

【0065】以上のようなループ表の作成を終了すると、続いて、コンパイラ108は、まず、ループ1を処理対象ループnとして取り出し(S204)、ループ1に対してソフトウェアプリフェッチ処理を行う(S205)。

【0066】そのため、まず、処理対象ループnであるループ1内の未処理の配列参照A(i)を処理対象配列参照sとして取り出す(S302)。また、変数Lに、ターゲットマシンのメモリレイテンシを代入する(S303)。

【0067】次に、ループ1が最内側ループかな否かを調

べる (S304)。ループ1は、内側ループとして、ループ2を有し、最内側ループではないので (S304: no)、次に、プリフェッチ命令を挿入すべき位置を求める (S307)。

【0068】そのため、まず、処理対象配列参照sであるA(i)の直前のループであるループ2を分割対象ループmとする (S401)。また、kにLを代入する (S402)。そして、分割対象ループmであるループ2の全体予測実行サイクル数を求める (S403)。この場合、ループ2の繰り返し回数Mは、コンパイル時には不明なので、 $d \leftarrow \infty$ とする。

【0069】そして、dがkより大きいかな否かを調べる (S404)。この場合、 $d (= \infty)$ は $k (= L)$ より大きいので (S404: yes)、現時点の分割対象ループmであるループ2の分割を行うため、ループ2のループ1回分の予測実行サイクル数cを求め、 $\beta = \lceil L/c \rceil$ を計算する (S405)。そして、ループ2を、 $1 \sim (M - \beta)$ 回を実行する前半ループm1と $(M - \beta + 1) \sim M$ 回を実行する後半ループm2とに分割する (S406)。

【0070】そして、後半ループm2が多重ループであるかな否かを調べる (S407)。この場合、後半ループm2は多重ループではないので (S407: no)、現時点での前半ループm1と後半ループm2の間をプリフェッチ命令を挿入すべき位置とする (S409)。そして、その位置にプリフェッチ命令「prefetch (A(i))」を挿入する (S308)。

【0071】図5(c)は、プリフェッチ命令を挿入した後のプログラムをソースコードイメージで示したものである。同図(c)に示すように、同図(a)のループ(605~607)が、始めの $(M - \beta)$ 回を実行する前半ループ(624~626)と終わりの β 回を実行する後半ループ(628~630)の2つに分割されて、その間に、配列参照A(i)のプリフェッチ命令(627)が挿入されている。

【0072】このようにすると、プリフェッチ命令(627)とデータ参照(631)の間に実行されるループは、 β 回しか回らず、 β は一般に元のループ回数Mに比べて小さいので、その間に参照されるデータ量は一般に小さく、プリフェッチしたデータがキャッシュから追い出される可能性は低くなる。また、ループの β 回の実行には、Lサイクル、すなわち、データをメモリからプリフェッチするのに要するのと等しいサイクル数が必要となるので、データを参照するときにプリフェッチデータの到着が間に合っていないということがない。

【0073】次に、コンパイラ108のコンパイル処理を、別の具体例に基づいて説明する。図6(a)は、コンパイル処理の対象とされる別のソースプログラムを示す図である。ここでは、配列参照A(i1)(811)に対してプリフェッチを行う場合を考える。

【0074】図6(b)は、同図(a)のプログラムに対して、コンパイラ108が作成するループ表110の構成例を示す図である。同図(a)のプログラムにはループが4つあり、ループ(802~812)がループ1、ループ(803~807)がループ2、ループ(804~806)がループ3、ループ(808~810)がループ4とされる。

【0075】また、ループ1のループ繰り返し変数は「i1」、繰り返し回数は「10」、内側ループは、ループ「2」、「4」、直前ループ、直後ループは、「なし」である。ループ2のループ繰り返し変数は「i2」、繰り返し回数は「20」、内側ループは、ループ「3」、直前ループは「なし」、直後ループは、ループ「4」である。ループ3のループ繰り返し変数は「i3」、繰り返し回数は「30」、内側ループ、直前ループ、直後ループは、いずれも「なし」である。ループ4のループ繰り返し変数は「i4」、繰り返し回数は「40」、内側ループ、直後ループはいずれも「なし」、直前ループは、ループ「2」である。

【0076】また、ループ3及びループ4のループ1回あたりの予測実行サイクルを、それぞれ、「5」及び「1」と見積り、ループ2のループ1回あたりの予測実行サイクルを「150」($= 5 * 30$)、ループ1のループ1回あたりの予測実行サイクルを、「3040」($= 150 * 20 + 1 * 40$)と算出している。

【0077】このようなループ表110の作成を終了すると、コンパイラ108は、まず、処理対象ループnとして、ループ1を取り出し (S204)、ループ1に対してソフトウェアプリフェッチ処理を行う (S205)。

【0078】そのため、まず、処理対象配列参照sとして、A(i1)を取り出す (S302)。また、Lにターゲットマシンにおけるメモレイテンシを代入する (S303)。ここでは、 $L = 100$ とする。次に、処理対象ループnであるループ1が最内側ループかな否かを調べる (S304)。ループ1は、内側ループとしてループ2、4を含み、最内側ループではないので、続いて、プリフェッチ命令を挿入すべき位置を求める (S307)。

【0079】そのために、まず、処理対象配列参照sであるA(i1)の直前のループ、すなわち、ループ4を分割対象ループmとする (S401)。またkに $L (= 100)$ を代入する (S402)。

【0080】次に、分割対象ループmであるループ4の全体予測実行サイクル数dを求める (S403)。ループ4のループ1回分の予測実行サイクルは、「1」で、ループ繰り返し回数が「40」なので、 $d \leftarrow 40 (= 1 * 40)$ となる。

【0081】次に、dがkより大きいかな否かを調べる (S404)。この場合、 $d (= 40)$ は、 $k (= 100)$

0)より小さいので (S404: no)、分割対象ループmを、現在の分割対象ループであるループ4の直前のループ、すなわち、ループ2とし、また、 $k \leftarrow 60 (=k-d=100-40)$ とする (S410)。

【0082】そして、改めて、現在の分割対象ループmであるループ2の全体予測実行サイクル数dを求める (S403)。ループ2のループ1回分の予測実行サイクルは「150」で、ループ繰り返し回数が「20」なので、 $d \leftarrow 3000 (=150 \times 20)$ となる。

【0083】次に、dがkより大きいかなかを調べる (S404)。この場合、 $d (=3000)$ は、 $k (=60)$ より大きいので、分割対象ループmであるループ2の分割を行うため、次に、ループ2のループ1回あたりの予測実行サイクル数e ($=150$) を求め、 $\beta = [k/e] = [60/150] = 1$ を計算する (S405)。次に、その結果に基づいて、ループ2を2つのループに分割する (S406)。ループ2の繰り返し回数は20、 $\beta = 1$ なので、ループ2を1～19回を実行する前半ループm1と、最後の1回を実行する後半ループm2に分割する。

【0084】図7(a)は、図6(a)に示したプログラムのループ2を、前半ループm1と後半ループm2に分割した時点でのプログラムをソースコードイメージで示したものである。同図に示すように、元のループ(803～807)が、前半ループ(903～907)と、後半ループ(908～912)に分割されている。

【0085】図7(b)は、ループ2の分割に伴って、コンパイラ108が更新した後のループ表110を示す図である。同図(a)に示すように、ループ2の分割によって、2つのループが増えているが、ループ(908～912)がループ5、ループ(909～911)がループ6とされる。

【0086】また、ループ5のループ繰り返し変数は「i2」、繰り返し回数は「1」、内側ループは、ループ「6」、直前ループは、ループ「2」、直後ループは、ループ「4」、予測実行サイクルは、「150」となる。ループ6のループ繰り返し変数は「i3」、繰り返し回数は「30」、内側ループ、直前ループ、直後ループは、いずれも「なし」、予測実行サイクルは、「5」となる。

【0087】更に、ループ1の内側ループに、ループ「5」が追加されており、ループ2の繰り返し回数及び直後ループがそれぞれ、「19」及びループ「5」に更新され、ループ4の直前ループが、ループ「5」に更新されている。

【0088】コンパイラ108は、ループ2の分割を終了すると、次に、後半ループm2であるループ5が多重ループであるかなかを調べる (S407)。この場合、ループ5は、内側ループとして、ループ6を含み、多重ループがあるので (S407: yes)、分割対象ループ

mを、ループ6に変える (S408)。

【0089】その後、改めて、分割対象ループmであるループ6の全体予測実行サイクルを求める (S403)。この場合、 $150 (=30 \times 5)$ であるので、それをdとする。次に、dがkより大きいかなかを調べる (S404)。この場合、 $d (=150)$ は、 $k (=60)$ より大きいので (S404: yes)、ループ6の分割を行うため、ループ6のループ1回あたりの予測実行サイクル数e ($=5$) を求め、 $\beta = [k/e] = [60/5] = 12$ を計算する (S405)。

【0090】そして、ループ6を最後の β 回のみを実行する後半ループm2と、その前までの繰り返しを実行する前半ループm1とに分割する (S406)。ループ6の繰り返し回数は30なので、ループ6を、1～18 ($=30-12$) 回実行する前半ループm1と、19～30回実行する後半ループm2に分割する。続いて、後半ループm2が多重ループかなかを調べると (S407)、そうではないので (S407: no)、現時点での前半ループm1と後半ループm2の間をプリフェッチ命令を挿入すべき位置とし (S409)、その位置に、処理対象配列参照sであるA(i1)についてのプリフェッチ命令を挿入する (S308)。

【0091】図8は、プリフェッチ命令を挿入した時点でのプログラムをソースコードイメージで示したものである。同図に示すように、前半ループ(1009～1011)と後半ループ(1013～1015)の間にプリフェッチ命令(1012)が挿入されている。なお、この場合、ループ(1008～1016)は1回しか回らないので、前述したように、ここはループにしなくてもよい。

【0092】図8のプログラムでは、配列要素A(i1)のプリフェッチ命令(1012)と配列要素A(i1)の参照(1020)の間で、ループ(1013～1015)とループ(1017～1019)が実行される。ループ(1013～1015)の予測実行サイクル数は、 $5 \times (30-19+1) = 60$ であり、ループ(1017～1019)の予測実行サイクル数は、 $1 \times 40 = 40$ である。つまり、合計100サイクルであり、メモリレイテンシと等しくなる。従って、データの参照にちょうど間に合うように、プリフェッチ命令が発行されることになる。

【0093】以上の実施形態では、プリフェッチ命令を挿入するためにループを分割していたが、ループを分割せず、ループ中に条件付きのプリフェッチ命令を挿入するようにしてもよい。

【0094】図9は、図5(a)に示したプログラムにおいて、ループを分割してプリフェッチ命令を挿入するかわりに、ループ中に条件付きのプリフェッチ命令実行コード(1106～1107)を挿入した場合のプログラムをソースコードイメージで示した図である。この場

合、制御変数 j が $(M-\beta+1)$ と等しいとき、すなわち最後から β 番めの繰り返しのときに $A(i)$ のプリフェッチ命令が実行され、プリフェッチ命令が実行されるタイミングは、図5(c)の場合と同じになる。このような処理を行うには、例えば、図4のステップS406でループを分割するかわりに、プリフェッチ命令を発行すべきループ m の繰り返し回数 $N-\beta+1$ を記憶しておいて、その後、その条件に該当するときに処理対象配列参照 s のプリフェッチ命令を実行するようなコードを挿入するようにすればよい。

【0095】また、以上の実施形態では、メモリ参照がループ内にある場合を対象にしていたが、ループ内がない場合に適用することもできる。図10は、プログラム手続き「subroutine f(A,B,i)」に対して、本発明を適用した例を示す図である。

【0096】ここでは、図10(a)に示したプログラム手続き内の配列参照 $A(i)$ (1207) に対して、同図(b)のように直前のループを分割してプリフェッチ命令(1215)を挿入している。このような処理によって、メモリレイテンシ分だけ前にプリフェッチ命令を実行することができる。このような処理を行うには、図2のステップS203～S204でループを取り出すとき、プログラム手続き全体を1つの仮想的なループと考慮して処理すればよい。

【0097】

【発明の効果】以上詳細に説明したように、本発明によれば、最内側ループ中にないデータの参照に対して、プリフェッチしたデータが使用される前にキャッシュから追い出されることに起因するキャッシュミスを減少させ、プログラム実行の高速化を図ることができる。

【図面の簡単な説明】

【図1】 本発明によるコンパイラが稼動する計算機システムの構成を示す図である。

【図2】 コンパイラ108のコンパイル処理の流れを示すフローチャートである。

【図3】 ソフトウェアプリフェッチ処理の流れを示す図である。

【図4】 プリフェッチ命令を挿入すべき位置を求める処理の流れを示す図である。

【図5】 本発明によるソフトウェアプリフェッチ処理の例を示す図である

【図6】 コンパイル処理の対象とされるプログラムの例とそのループ表を示す図である。

【図7】 ループ分割をした時点でのプログラムのソースコードイメージとそのループ表を示す図である。

【図8】 プリフェッチ命令を挿入した時点でのプログラムをソースコードイメージで示した図である。

【図9】 ループ中に条件付きのプリフェッチ命令実行コードを挿入した場合のプログラムをソースコードイメージで示した図である。

【図10】 プログラム手続き内の配列参照に対するソフトウェアプリフェッチ処理の例を示す図である。

【図11】 従来のソフトウェアプリフェッチ方法の例を示す図である。

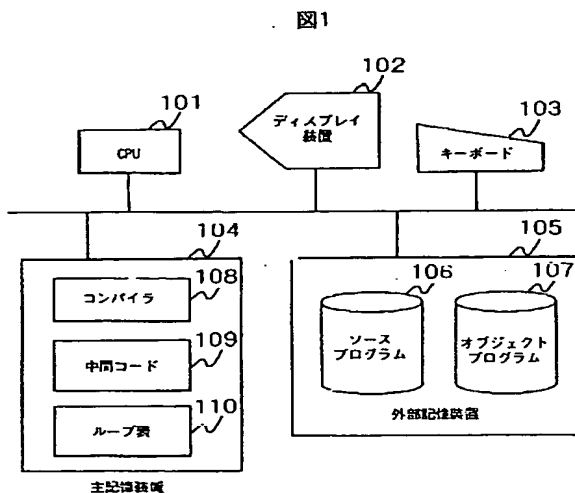
【図12】 従来のソフトウェアプリフェッチ方法の例を示す図である

【符号の説明】

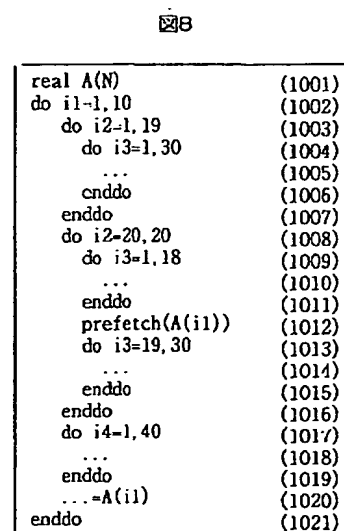
108 コンパイラ

S205 ソフトウェアプリフェッチ処理

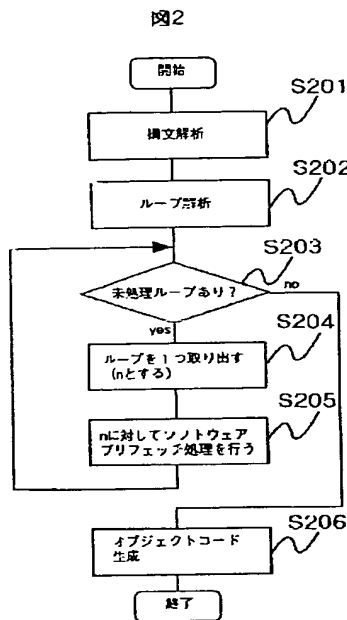
【図1】



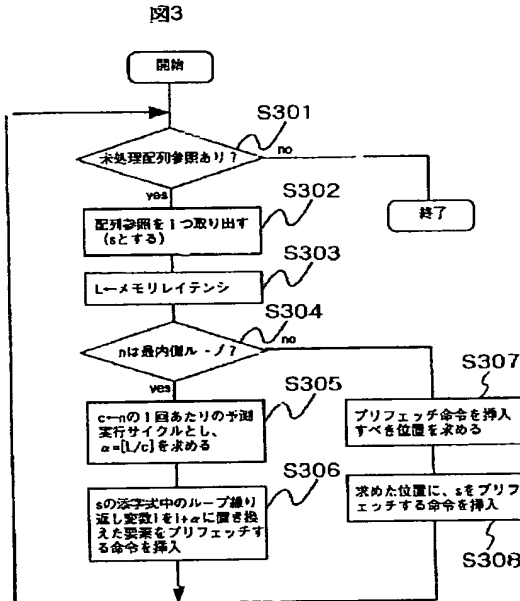
【図8】



【図2】



【図3】



【図9】

```

real A(N),B(M,N) (1101)
s=0 (1102)
do i=1,N (1103)
  t=0 (1104)
  do j=1,M (1105)
    if j=M-β+1 then (1106)
      prefetch(A(i)) (1107)
    t=t+B(j,i) (1108)
  enddo (1109)
  s=t+A(i) (1110)
enddo (1111)
  
```

【図5】

【図4】

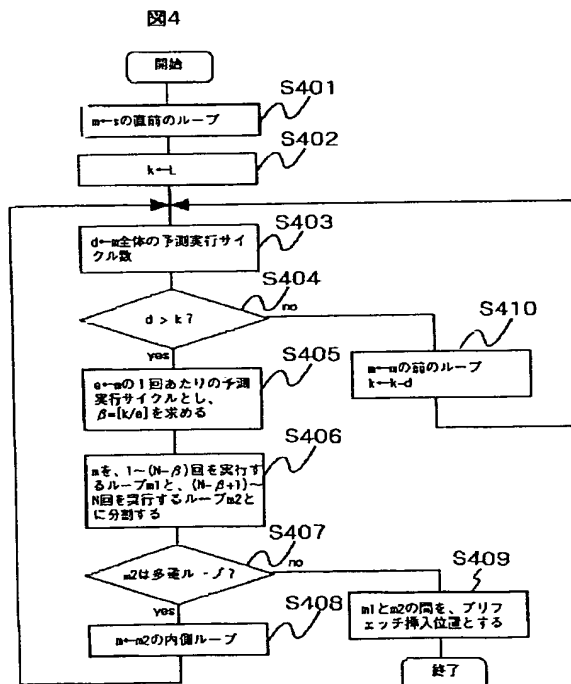


図5

```

real A(N),B(M,N) (601)
s=0 (602)
do i=1,N (603)
  t=0 (604)
  do j=1,M (605)
    t=t+B(j,i) (606)
  enddo (607)
  s=t+A(i) (608)
enddo (609)
  
```

(a)

ループ 番号	ループ繰り返し 変数	繰り返し 回数	内側ループ	直線ループ	直後ループ	予測実行 サイクル
1	i	N	2	なし	なし	c*M
2	j	M	なし	なし	なし	c

(b)

```

real A(N),B(M,N) (620)
s=0 (621)
do i=1,N (622)
  t=0 (623)
  do j=1,M-β (624)
    t=t+B(j,i) (625)
  enddo (626)
  prefetch(A(i)) (627)
  do j=M-β+1,M (628)
    t=t+B(j,i) (629)
  cnddo (630)
  s=t+A(i) (631)
enddo (632)
  
```

(c)

【図6】

図6

```

real A(N)          (801)
do i1=1,10         (802)
  do i2=1,20       (803)
    do i3=1,30     (804)
      ...          (805)
    enddo          (806)
  enddo           (807)
do i4=1,40        (808)
  ...            (809)
enddo             (810)
...=A(i1)         (811)
enddo             (812)

```

(a)

ループ 番号	ループ繰り返し 変数	繰り返し 回数	内側ループ	直前ループ	直後ループ	予測実行 サイクル
1	i1	10	2,4	なし	なし	3040
2	i2	20	3	なし	4	150
3	i3	30	なし	なし	なし	5
4	i4	40	なし	2	なし	1

(b)

【図10】

図10

```

subroutine f(A,B,i) (1201)
real A(N),B(M,N)  (1202)
t=0               (1203)
do j=1,M          (1204)
  t=t+B(j,i)      (1205)
enddo             (1206)
t=t+A(i)          (1207)
end               (1208)

```

(a)

```

subroutine f(A,B,i) (1209)
real A(N),B(M,N)  (1210)
t=0               (1211)
do j=1,M-β        (1212)
  t=t+B(j,i)      (1213)
enddo             (1214)
prefetch(A(i))    (1215)
do j=M-β+1,M      (1216)
  t=t+B(j,i)      (1217)
enddo             (1218)
t=t+A(i)          (1219)
end               (1220)

```

(b)

【図11】

図11

```

real A(N)          (501)
s=0                (502)
do i=1,N           (503)
  s=s+A(i)         (504)
enddo              (505)

```

(a)

```

real A(N)          (506)
s=0                (507)
do i=1,N           (508)
  prefetch(A(i+α)) (509)
  s=s+A(i)         (510)
enddo              (511)

```

(b)

【図7】

図7

```

real A(N)          (901)
do i1=1,10         (902)
  do i2=1,19       (903)
    do i3=1,30     (904)
      ...          (905)
    enddo          (906)
  enddo           (907)
do i2=20,20        (908)
  do i3=1,30       (909)
    ...            (910)
  enddo           (911)
enddo             (912)
do i4=1,40         (913)
  ...            (914)
enddo             (915)
...=A(i1)         (916)
enddo             (917)

```

(a)

ループ 番号	ループ繰り返し 変数	繰り返し 回数	内側ループ	直前ループ	直後ループ	予測実行 サイクル
1	i1	10	2,4,5	なし	なし	3040
2	i2	19	3	なし	5	150
3	i3	30	なし	なし	なし	5
4	i4	40	なし	5	なし	1
5	i2	1	6	2	4	150
6	i3	30	なし	なし	なし	5

(b)

【図12】

図12

```

real A(N),B(M,N)  (601)
s=0               (602)
do i=1,N          (603)
  t=0             (604)
  do j=1,M        (605)
    t=t+B(j,i)    (606)
  enddo           (607)
  s=t+A(i)        (608)
enddo             (609)

```

(a)

```

real A(N),B(M,N)  (610)
s=0               (611)
do i=1,N          (612)
  t=0             (613)
  do j=1,M        (614)
    t=t+B(j,i)    (615)
  enddo           (616)
  prefetch(A(i+α)) (617)
  s=t+A(i)        (618)
enddo             (619)

```

(b)

THIS PAGE BLANK (USPTO)